Praxen Freier Software.Wie Freie Softwareprojekte Funktionieren

Seminararbeit im Rahmen des Seminars

„Wissen in der modernen Gesellschaft“

Universität Leipzig, Sommersemester 2009

Verfasser: Yuan Zhang (9414466)

Betreuer: Prof. Hans-Gert Gräbe

11.6.2009

0. Vorwort

,,Freiheit" schreit Mel Gibson alias William Wallace am Ende des Hollywood-Epos Braveheart, bevor sein Kopf von den Soldaten der englischen Armee auf der Folterbank vom Körper getrennt wird. Der schwer kranke, dahin scheidende englische König und Unterdrücker des schottischen Volks nimmt diesen Ruf mit in den Tod und erkennt trotz seiner Eroberungen die Stärke und Unbrechbarkeit des menschlichen Geistes.

Hat diese Szene symbolischen Charakter bezüglich der Softwareindustrie? Nun, es gibt zwar kein vergleichbares Martyrium wie das des schottischen Nationalhelden und (glücklicherweise) keine anderen Todesfälle, aber ein Kampf um Marktanteile (anstelle von Territorien) findet dennoch statt, und in diesem Kampf spielt Freiheit eine bedeutende Rolle. Es ist die Freiheit eines jeden gemeint, Software zu benutzen, zu verändern, zu vervielfältigen und weiter zuverbreiten. Richard Stallman, Vater der freien Software-Bewegung und seine um ihn versammelte Gemeinde, sehen das als ihr ureigenes Menschenrecht an.

Freie Software gibt es schon solange wie Software selbst. Fast jeder arbeitet (unbewusst) mit ihr, wenn er im Internet surft, E-Mails schreibt oder Beiträge in Newsgroups liest. Schub für den Bekanntheitsgrad von freier Software war sicherlich das Betriebssystem Linux, das einen sagenhaften Aufstieg in der IT-Branche verzeichnen konnte.[1]

In dieses Kapitel soll dem Leser ein grundsätzliches Verständnis für freie Software vermitteln. Dazu dient der folgende Abschnitt ,,Was ist freie Software", ein Vergleich von freier Software mit anderen Softwarevarianten.Im Gegensatz zu teuren, kommerziellen und oftmals proprietären Programmen ist freie Software kostenlos, wird gemeinsam von Entwicklern aus verschiedenen Kontinenten dieser Erde programmiert, die sich noch nicht einmal kennen und ist trotzdem stabil und zuverlässig. Wie kann das sein? Gibt es ein Entwicklungsmodell, und wenn ja, wie funktioniert es? Welche erfahren gibt es,wenn Konflikte oder Bugs auftreten? Danach wird der mögliche Softwarezyklus einer freien Software genauer beleuchtet, ihre Eigenschaften erläutert und abschließend auf die Einsatzgebiete freier Software eingegangen.

[1] Bravehack, Jens Sieckmann,Url: http://www.bravehack.de/html/bravehack.html

1 . Grundlagen

In diesem Kapitel wird zum Anfang erklärt was freie Software ist. Danach werden Grundeigenschaften sowie die Leistungsfähigkeit von freier Software einleiten vorgestellt.

1.1 Was ist freie Software

Freie Software (engl.: *free software*) ist Software, die für jeden Zweck genutzt, studiert, bearbeitet und in ursprünglicher oder veränderter Form weiterverbreitet werden darf. Das schließt auch die kommerzielle Nutzung ein. Software, die auf Grundlage dieser Software entsteht, muss normalerweise ebenfalls freie Software sein (Prinzip des Copyleft bzw. *share alike*).[2]

Freier Software steht der proprietäre oder **„**unfreie**“** Software gegenüber, die diese Freiheiten nicht oder nicht in vollem Umfang bietet. Diese Unterscheidung wurde von der Free Software Foundation (FSF) geprägt.

1.2 Grundeigenschaften freier Software

Alle weiteren Merkmale von freier Software ergeben sich aus ihren vier Grundeigenschaften: **(1)** die Software darf ohne Einschränkungen benutzt werden, **(2)** der Quellcode freier Software ist verfügbar, er darf studiert und aus ihm darf gelernt werden, **(3)** die freie Software darf ohne Einschränkungen und ohne Zahlungsverpflichtungen kopiert und weitergegeben werden, **(4)** Veränderungen an der Software dürfen durchgeführt und weitergegeben werden.

Diese Eigenschaften bilden die idealen Voraussetzungen für eine offene, d.h., nicht auf Arbeitsvertragsverhältnissen beruhende, kooperative Softwareentwicklung und eine weitestgehende Beteiligung der Anwender.

Jedes größere Softwareprojekt wird von Gruppen von Entwicklern erstellt. Auch in der Industrie hat man heute Teams, die über eine kreative Selbstständigkeit verfügen, die ihren Code synchronisieren und in regelmäßigen Abständen das gemeinsame Produkt stabilisieren. In denen hierarchische Verfahren unter einem Chefprogrammierer weitgehend abgelöst sind[3]

1.3 Die Leistungen freier Software

[2] http://de.wikipedia.org/wiki/Freie_Software

[3] Vgl. Grassmuck 2004, S. 233

70% aller Webserver laufen auf dem freien Apache. Qualität und Zuverlässigkeit freier Software ist meist höher als bei vergleichbaren kommerziellen Produkten.

- Wikipedia - eine Freie Enzyklopädie: Wikipedia hat sich zum Ziel gesetzt eine freie Enzyklopädie zu erstellen. Wikipedia hat seit 2001 etwa 575.000 Artikel im englischsprachigen Wikipedia und etwa 240.000 Artikel im deutschsprachigen Teil (Stand Mai 2005). Insgesamt gibt es etwa 1,5 Millionen Artikel in 195 Sprachen. Die Artikel können dabei jederzeit anonym verändert werden, dennoch ist die Qualität der Artikel vergleichbar oder zum Teil höher als bei kommerziellen Enzyklopädien. Im Faktor „Aktualität“ ist Wikipedia jedenfalls nicht zu schlagen. Wikipedia Artikel unterliegen der GFDL (GNU Free Documentation Licence) die ähnlich wie GPL bei Software sicherstellt, dass die Artikel kein Privateigentum der Autoren mehr sind.[4]

[4] http://de.wikipedia.org/wiki/Wikipedia:Hauptseite

2. Quellcode und Objektcode

In diesem Abschnitt werde die Begriffe Quellcode und Objektcode vorgestellt. Sind diese beiden Begriffe erklärt wird auf die Veränderungsfreiheit von Software eingegangen. Dabei werden die Punkte warum es sinnvoll ist Veränderungsfreiheit von Software zuzulassen und für welche Nutzer dies nützlich sein kann.

2.1 Code

Software stellt eine besondere Klasse von Wissen dar. Es handelt sich um operative Anweisungen, die sich, von Menschen geschrieben, an eine Computer richten (Quellcode). Menschen schreiben diese Anweisungen in eine höheren Programmiersprache, wie Pascal, C, oder C++. Dieser Quelltext wird einem Compiler übergeben, einem Softwarewerkzeug, das ihn in eine maschinennähere, für Menschen nicht mehr lesbare Form, den Objektcode übersetzt. Diese binäre Form des Programms erst veranlasst den Rechner, für den Menschen (meist) sinnvolle Zeichenoperationen auszuführen.[5]

Die Begriffe sind intuitiv, der Quellcode ist das Mittel zum Zweck. Mit anderen Worten, der Quellcode ist der Anfang, oder die Quelle. Der Betrieb und der Objektcode ist das gewünschte Ergebnis, oder das Objekt, die gesamte Übung. Objektcode wird in Dateien gespeichert, der aus dem Quellcode durch den Compiler generiert wird. Die Generierung des Objektcodes kann auch als Ende der Arbeit für den Programmierer bedeuten.[6]

2.2 Veränderungsfreiheit

Quellcode ist Voraussetzung für Veränderungsfreiheit. Für reine Anwender mag eine Blackbox zunächst keine wirkliche Qualitätseinschränkung bedeuten, für Entwickler und Systemadministratoren macht es ihre Arbeitsumgebung jedoch zu einer Welt voller Mauern und verbotener Zonen. Für Anpassung, Integration, Fehlerbehebung und Ergänzung müssen sie Veränderungen an der Software vornehmen, und das ist nur am Quelltext möglich. Freie, quell offene Software bietet ihnen diese Möglichkeit. Bei proprietärer Software bleibt ihnen nur, auf die Unterstützung durch den Hersteller zu hoffen oder sich durch umständliches und nur in engen Grenzen legales Reserve Engineering zu behelfen.

Ein häufig von Anbietern proprietärer Software vorgebrachtes Argument lautet, ihre Software umfasse mehrere Millionen Zeilen Code, weshalb ihre Kunden gar kein

[5] Vgl. Grassmuck 2004, S. 233

[6] http://wasista.de/Internat/2009/0225/Was-ist-Objekt-Code-in-einem-Computer-Pr.html

Interesse hätten, den Quellcode zu lesen. Im übrigen verfalle durch einen Eingriff in die Software die Gewährleistungsgarantie, und der Support würde ungleich komplexer. Das Gegenargument lautet, dass keine Software 100-prozentig das leistet, was Anwender sich wünschen. Außerdem veraltet Software rasch, wenn sie nicht ständig den sich schnell verändernden Soft- und Hardwaregegebenheiten angepasst wird. Anspruchsvolle Anwender werden also in jedem Falle die Möglichkeit schätzen, in den Quellcode eingreifen oder andere damit beauftragen zu können, die gewünschten Veränderungen für sie vorzunehmen. Selbst wenn keine Änderungen vorgenommen werden sollen, ist es sinnvoll, den Quellcode mit zuliefern, da Interessierte ihn lesen und daraus lernen können.[7]

[7] Vgl. Grassmuck 2004, S. 233

3. Wie funktioniert ein Projekt der freien Software

Es beginnt meist damit, dass jemand ein Problem hat. [8]

Das Sprichwort „Notwendigkeit ist die Mutter aller Erfindungen“ übersetzt Eric Raymond in eine der Faustregeln der freien Software: „Jedes gute Softwarewerk beginnt damit, dass ein Entwickler ein ihn persönlich betreffendes Problem angeht“ (Raymond, 1998).

Freie Software entsteht also zunächst nicht auf Anweisung eines Vorgesetzten oder Auftraggebers. Sie ist vielmehr eine eigen motivierte Tätigkeit, angetrieben von dem Wunsch, ein auf der Hand liegendes Problem bei der Arbeit oder Forschung zu lösen. Eine Ausnahme bildet das GNU-Projekt, das von der Vision eines vollständigen freien Betriebssystems geleitet wurde.[9]

All diese Männer der ersten Stunde, es scheint tatsächlich nicht eine einzige Frau unter ihnen zu geben, veröffentlichten ihre Projekte frühzeitig, in der Annahme, dass sie nicht die einzigen sind, die dieses Problem haben, und dass es andere gibt, die ihnen bei der Lösung helfen würden.[10]

- David Harris arbeitete an einer Universität in Dunedin,die 1989 ein Novell NetWare-Netzwerk installierte, nur um festzustellen, dass es kein Emailsystem enthielt. Die kommerziellen E-Mailpakete waren sehr teuer, also schrieb Harris in seiner Freizeit ein einfaches Programm namens Pegasus Mail, das er seither zu einem mächtigen und komfortablen Werkzeug weiterentwickelt hat und weiterhin verschenkt.[11]

8 Vgl. Grassmuck 2004, S. 235
9 Vgl. Grassmuck 2004, S. 236
10 http://www2.dortmund.de/do4u_intern/artnet_archiv/ger/wor/grassm.html#N_19
11 Vgl. Grassmuck 2004, S. 236

4. Core-Team und Maintainer

Da nun bekannt ist was Quellcode und Objektcode ist und wie ein freies Projekt funktioniert werden nun die Mitglieder vorgestellt die funktionierende Versionen der Software zur Verfügung stellen. Es wird erklärt was das Core-team ist und was Maintainer für aufgaben haben.

4.1 Core-Team

Wenn die Zahl der Mitentwickler und Anwender wächst, bildet diese Gruppe das zentrale Steuerungsgremium des Projekts, das *Core-Team.* Ein solches Team rekrutiert sich meist aus den Leuten, die entweder schon am längsten daran arbeiten oder sehr viel daran gearbeitet haben oder derzeit am aktivsten sind. Innerhalb des Core-Team werden Entscheidung über die allgemeine Richtung der Entwicklung gefällt, über Fragen des Designs und über interessante Probleme an denen weitergearbeitet werden soll. Große Projekte werden in funktionale Einheiten, in "Packages" oder Module gegliedert, für die jeweils ein oder mehrere Maintainer zuständig sind.[12]

- XFree86 hat ein Core-Team von elf Personen und eine Community von etwa 600 Entwickler.

- Das Apache-Projekt umfasst 22 Leute aus sechs Ländern.

4.2 Maintainer

Die Rolle eines Maintainers ist es zunächst, Ansprechpartner für die jeweilige Software zu sein. Häufig handelt es sich um altgediente Coder, die sich eine Reputation erworben haben. Als treibende Kraft wirkt die Gemeinschaft. An jedem Einzelprojekt arbeiten mehrere Dutzend bis hundert Entwickler weltweit mit. Änderungen werden an das Core-Team geschickt und von diesem in den Quellcode integriert.

- Beim Linux-Kernel gibt es kein offizielles Entwicklerteam. Im Laufe der Zeit hat sich meritokratisch eine Gruppe von fünf oder sechs Leuten heraus geschält, die das Vertrauen des zentralen Entwicklers Linus Torvalds genießen. Sie sammeln die eintreffenden Patches, die Code-Flicken, sortieren Unsinniges und Unfertiges aus, testen den Rest und geben ihn an Torvalds weiter. Er trifft dann aufgrund der Vorentscheidungen seiner Vertrauten die letzte Entscheidung (vgl. Dietrich, 1999).[13]

[12] Vgl. Grassmuck 2004, S. 237
[13] Vgl. Grassmuck 2004, S. 237

5. Die Community

*Jeder macht ,wozu er **Lust** hat.....*

Die neueren Projekte, wie Linux und Apache messen der Entwicklergemeinschaft einen grösseren Stellenwert bei, als das ältere GNU-Projekt. Im GNU Manifest von 1985 schrieb Stallman: „GNU ... ist ... das Softwaresystem, das ich schreibe, um es frei weiterzugeben... Mehrere andere Freiwillige helfen mir dabei“ (Stallman, 1985). In einem Interview im Januar 1999 antwortete er auf die Frage, wie viele Menschen im Laufe der Jahre an GNU mitgearbeitet haben: „Es gibt keine Möglichkeit, das festzustellen. Ich könnte vielleicht die Anzahl der Einträge in unserer Freiwilligendatei zählen, aber ich weiß nicht, wer am Ende wirklich etwas beigetragen hat. Es sind wahrscheinlich Tausende. Ich weiß weder, wer von ihnen echte Arbeit geleistet hat, noch ist jeder, der echte Arbeit geleistet hat, dort aufgeführt. Bei den jüngeren Projekten wird stärker auf das im übrigen auch Urheberpersönlichkeitsrecht verbriefte, Recht auf Namensnennung geachtet. Der Pflege der Community, die die Basis eines Projektes bildet, wird mehr Aufmerksamkeit gewidmet. Das Debian-Team besteht aus etwa 500 Mitarbeitern weltweit. Bei XFree86 sind es rund 600. Bei den meisten freien Softwareprojekten gibt es keine festgelegte Aufgabenverteilung. Jeder macht das, was ihn interessiert und programmiert oder implementiert das, wozu er Lust hat. Auch der Apache-Webserver wird natürlich nicht allein von den 22 Core-Team-Mitgliedern entwickelt. Es gibt sehr viele Leute, die regelmäßig, gelegentlich oder zum Teil auch nur einmalig Fehler im Apache beheben. Die Beteiligung fängt bei simplen Fehlerberichten (*Bug Reports*) an und geht über Vorschläge für neue Funktionen (*Feature Requests*) und Ideen zur Weiterentwicklung bis hin zu *Patches* oder größeren Funktionserweiterungen, die von Nutzern erstellt werden, die ebenfalls Spaß am Entwickeln haben und ihren Teil dazu beitragen wollen, den Apache zu verbessern.[14]

[14] Vgl. Grassmuck 2004, S. 238

6. rough concensus and running code

In diesem Abschnitt wird die Philosophie von freien Softwareprojekten erklärt und was die Grundsätze sind. Danach werden die Begriffe rough concensus und running code eingeführt.

6.1 Philosophie in den freien Softwareprojekten

„Wir wollen keine Könige, Präsidenten und Wahlen. Wir glauben an einen groben Konsens und an ablauffähigen Code.“ (Das Credo der Internet-Entwicklergemeinde)

Die gleiche Philosophie herrscht auch in den meisten freien Softwareprojekten. Auch die Core-Team-Mitglieder sind keine Projektleiter oder Chefs. Lars Eilebrecht über den Apache, der mit Abstimmungsverfahren und Widerspruchsfreiheitspflicht im Core-Team unter den freien Projekten mit über die formalisiere Entscheidungsverfahren verfügt:

- *„Alle Entscheidungen, die getroffen werden, sei es nun welche Patches, welche neuen Funktionalitäten in den Apache eingebaut werden, was in der Zukunft passieren soll und sonstige Entscheidungen werden alle auf Basis eines Mehrheitsbeschlusses getroffen“*[15]

6.2 rough concensus

rough concensus = Grober Konsens

Ein grober Konsens ist erreicht, wenn eine relativ große Mehrheit der Mitglieder,die an der Abstimmung teilgenommen haben, eine Entscheidung dafür oder dagegen getroffen haben.[16]

6.3 running code

running code = ablauffähige Code

Runningcode ist die Implementation neuer Technologien statt nur die Beschreibung durch die Mitglieder und Community.

[15] Vgl. Grassmuck 2004, S. 239

[16] http://memory-alpha.org/de/wiki/Memory_Alpha:Richtlinien_und_Empfehlungen

7. Code-Forking

Das eben Gesagte bedeutet nicht, dass Konflikte prinzipiell nicht dazu führen können, dass Fraktionen getrennte Wege gehen. Tatsächlich hat es verschiedentlich Spaltungen von Projekten gegeben, bei denen sich mit der Entwicklergruppe natürlich auch die Code-Basis verzweigt (von engl. *fork,* die Gabel). Im schlimmsten Fall bedeutet dies den Tod eines Projekts, oder es entstehen daraus zwei ähnliche Projekte, die um Entwickler und Anwender konkurrieren dadurch verdopplen sich die Gesamtbemühungen für das ehemalige Projekt. Im günstigsten Fall kann eine Spaltung fruchtbar sein. Die entstehenden Einzelprojekte entwickeln die Software für verschiedene komplementäre Anwendungsschwerpunkte weiter, ergänzen sich und profitieren von Innovationen in den anderen Projekten. Die Möglichkeit, sich jederzeit von einem Projekt abzusetzen und es in eine eigene Richtung weiter zutreiben, wird auch als heilsam erachtet. Sie verhindert, dass nicht zu wartende Mammutprogramme entstehen und Personen, die Verantwortung für Programme tragen, sich nicht mehr darum kümmern. Vor allem steueren sie den Prozess: Wenn die Entwicklung am Bedarf von ausreichend vielen vorbei geht, kommt es irgendwann zur Verzweigung.[17]

[17] Vgl. Grassmuck 2004, S. 240

8. Die Werkzeuge

Die zentralen Kommunikationsmittel für die weltweit ortsverteilte Kooperation sind E-Mail, genauer Mailinglisten sowie Newsgroups. Für Echtzeitkommunikation verwenden einige Projekte auch den Internet Relay Chat (IRC).

Die Projekte präsentieren sich und ihre Ressourcen auf Websiten. Das zentrale Instrument zur kooperativen Verwaltung des Quellcode sind CVS-Server. Das *Concurrent Versions System* (*CVS)* ist ein mächtiges Werkzeug für die Revisionsverwaltung von Softwareprojekten, das es Gruppen von Entwicklern erlaubt, gleichzeitig an denselben Dateien zu arbeiten, sich zu koordinieren und einen Überblick über die Veränderungen zu behalten. CVS ist Standard bei freier Software, aber auch viele Firmen, die proprietäre Software erstellen, setzen es ein.[18]

[18] Vgl. Grassmuck 2004, S. 241

9. Debugging

In diesem Abschnitt wird gezeigt wie mit Fehlern in freien Softwareprojekten umgegangen wird. Wie werden Fehler gemeldet, behoben und wie schnell werden Fehler behoben.

9.1 Bugs

Ein Programmfehler oder Softwarefehler, häufig auch Bug benannt, bezeichnet ganz allgemein ein Fehlverhalten von Computerprogrammen. Er tritt auf, wenn der Programmierer einen bestimmten Zustand in der Programmlogik nicht berücksichtigt hat, oder wenn die Laufzeitumgebung an sich fehlerhaft arbeitet. Auch Unvollständigkeit, Ungenauigkeiten, Mehrdeutigkeiten o. ä. in der Spezifikation des Programms können zu Fehlern führen, bzw. als solche interpretiert werden.

Computerprogramme enthalten in der Regel Programmfehler, die sich auch bei sorgfältiger Programmierung nicht vermeiden lassen. Erfahrungen aus der Softwaretechnik besagen, dass je 1000 Zeilen Code im Mittel etwa zwei bis drei Fehler enthalten. Um Fehler in Programmen aufdecken zu können, gibt es Programme, sogenannte Debugger, mit denen man ein Programm, welches man analysiert, schrittweise ablaufen und sich dabei sämtliche internen Zustände (Variablen) des Programms anzeigen lassen kann.[19]

9.2 Verfahren von Proprietärer Software

Fehler werden nicht versteckt**.** Bei Bugs herrscht, was Neal Stephenson eine institutionelle Unehrlichkeit nennt.

- „Kommerzielle Betriebssysteme müssen die gleiche offizielle Haltung gegenüber Fehlern einnehmen, wie sie die kommunistischen Länder gegenüber der Armut hatten. Aus doktrinären Gründen war es nicht möglich zuzugeben, dass Armut in den kommunistischen Ländern ein ernstes Problem darstellte, weil der Kommunismus sich ja gerade zum Ziel gesetzt hatte, die Armut zu beseitigen. Eben sowenig können kommerzielle Betriebssystemhersteller wie Apple und Microsoft vor aller Welt zugeben, dass ihre Software Fehler enthält und ständig abstürzt. Das wäre so, als würde Disney in einer Presseerklärung bekanntgeben, dass Mickey Mouse ein Schauspieler in einem Kostüm ist... Kommerzielle Betriebssystemanbieter sind als direkte Folge ihrer kommerziellen Ausrichtung dazu gezwungen, die äußerst unredliche Haltung einzunehmen, dass es sich bei Bugs um selten auftretende Abweichungen handelt, meist die Schuld von

[19] http://de.wikipedia.org/wiki/Programmfehler

anderen, und es daher nicht wert sind, genauer über sie zu sprechen“ (Stephenson, 1999).[20]

9.3 Verfahren von Freien Software

Freie Softwareprojekte dagegen können sich dem Problem offen stellen. Ihr Verfahren, mit Bugs umzugehen, stellt einen der wichtigsten Vorteile gegenüber proprietärer Software dar. Die Stabilität dieser Software, d.h. ihr Grad an Fehlerfreiheit, verdankt sich nicht der Genialität ihrer Entwickler,sondern der Tatsache, dass jeder Anwender Fehler an den Pranger stellen kann und die kollektive Intelligenz von Hundertern von Entwicklern meist sehr schnell eine Lösung dafür findet. Wer z.B. in GNU/Linux einem Bug begegnet sollte einen E-Mailbericht an die Entwickler schicken.

Dadurch können sich Tausende von Nutzern am Auffinden von Bugs und Hunderte von Entwicklern an ihrer Lösung beteiligen. Das Debugging kann hochgradig parallelisiert werden, ohne dass der positive Effekt durch erhöhten Koordinationsaufwand und steigende Komplexität konterkariert würde.

In der Formulierung von Raymond lautet diese Faustregel: „Wenn genügend Augen sich auf sie richten, sind alle Fehler behebbar“ (Raymond, 1998). Gemeint ist nicht etwa, dass nur triviale Fehler auf diese Weise beseitigt werden, sondern dass durch die große Zahl von Beteiligten die Chance steigt, dass einige von ihnen genau in dem fraglichen Bereich arbeiten und relative mühelos Lösungen finden können.[21]

Dieses Verfahren nicht nur durch seine Geschwindigkeit den Verfahren in der proprietären Software überlegen ist, sondern auch durch die Gründlichkeit seiner Lösungen:

- *„Dadurch, dass die Sourcen offen sind, werden oft genug nicht nur die Symptome behoben, wie wir das z. B. bei Windows-NT sehen, wo ein Problem im Internet-Information-Server ist und sie sagen, wir haben dafür jetzt gerade keinen Fix, weil das Problem tief im Kernel verborgen liegt und da können wir jetzt gerade nichts tun, sondern wir geben euch mal einen Patch, der verhindert, dass die entsprechenden Informationen bis zum Kernel durchdringen – also: Pflaster drauf kleben auf die großen Löcher im Wasserrohr. Das passiert halt in der Regel bei Open Source-Systemen nicht, da werden die Probleme tatsächlich behoben.“*[22]

[20] Vgl. Grassmuck 2004, S. 242
[21] Vgl. Grassmuck 2004, S. 243
[22] Vgl. Grassmuck 2004, S. 244

10. Die Releases

Durch die beschleunigten Innovationszyklen in der Informations- und Kommunikationstechnologie und hier besonders in der Software, werden Produkte in immer kürzeren Abständen durch neue Versionen obsolet gemacht. Nirgends ist diese Beschleunigung deutlicher als in der Informations- und Kommunikationstechnologie und hier besonders in der Software zu beobachten. Offizielle Versionen werden dann freigegeben (released), wenn sie den gewünschten Grad an Stabilität erreicht haben. Das Ausmaß der Änderung lässt sich an den Versionsnummern ablesen. Zum Beispiel Version 2 auf Version 3 markiert ein ganz neues Design. Version 3.3.1 zu Version 3.3.2 stellt einen kleineren Integrationsschritt dar.

Freie Software wird heute, wie mehrfach angesprochen, nicht als Produkt, sondern als kontinuierlicher Prozess verstanden. Zahlreiche Projekte haben bewiesen, dass eine Entwicklergemeinschaft ihre Software über lange Zeit zuverlässig und in hoher Qualität unterstützen und weiterentwickeln kann. Abhängig von der Geschwindigkeit werden täglich oder im Abstand von Wochen oder Monaten neue Releases herausgegeben.[23]

[23] Vgl. Grassmuck 2004, S. 246

11. Die Motivation: Wer macht freie Software

Was ist die Motivation der Community Zeit für ein freies Softwareprojekt zu investieren? Im Folgenden ist eine Liste mit Motivationsgründen gegeben:

- Selbsttätiges Schaffen als Wert an sich
- Freie Zeit für freie Software
- Die Lust des Lernens
- Kritik und Bestätigung von der Community
- Selbstmotivation und Geld verdienen schließen sich nicht aus
- Lieber freischaffend als angestellt

Unsere Gesellschaften kennen wohltätige, karitative und kulturelle (z.B. Kunstvereine) Formen des Engagements für eine gute Sache, die nicht auf pekuniären Gewinn zielen. Gemeinnützige Tätigkeiten sind steuerlich begünstigt und sozialwissenschaftlich in Begriffen wie *Civil Society*, Dritter Sektor und NGOs reflektiert. Doch während einer Ärztin, die eine Zeit lang in Myanmar oder Äthiopien arbeitet, allgemeine Anerkennung gezollt wird, ist die erste Reaktion auf unbezahlte Software-Entwickler meist, daß es sich entweder um Studenten handeln muß, die noch üben, oder um Verrückte, da ihre Arbeit in der Wirtschaft äußerst gefragt ist und gut bezahlt würde.

"Jedes Geschäft -- welcher Art es auch sei -- wird besser betrieben, wenn man es um seiner selbst willen als den Folgen zuliebe treibt", weil nämlich "zuletzt für sich Reiz gewinnt", was man zunächst aus Nützlichkeitserwägungen begonnen haben mag, und weil "dem Menschen Tätigkeit lieber ist, als Besitz, ... insofern sie Selbsttätigkeit ist".Diese Humboldtsche Erkenntnis über die Motivlage der Menschen bietet einen Schlüssel für das Verständnis der freien Software.

Ihre Entwicklung gründet in einem kreativen Akt, der einen Wert an sich darstellt. Ein *Learning-by-Doing* mag noch von Nützlichkeitserwägungen angestoßen werden, das Ergebnis des Lernens ist nicht nur ein individuelles Verständnis, sondern eine Schöpfung, die man mit anderen teilen kann. Statt also aus dem Besitz und seiner kommerziellen Verwertung einen Vorteil zu ziehen, übergeben die Programmiererinnen und Programmierer der freien Software ihr Werk der Öffentlichkeit, auf daß es bewundert, kritisiert, benutzt und weiterentwickelt werde. Die Anerkennung, die ihnen für ihre Arbeit gezollt wird, und die Befriedigung, etwas in den großen Pool des Wissens zurückzugeben, spielen sicher eine Rolle. Die wichtigste Triebkraft vor allen hinzukommenden Entlohnungen ist die kollektive Selbsttätigkeit. [24]

[24] http://www2.dortmund.de/do4u_intern/artnet_archiv/ger/wor/grassm.html#_1_7

12. Softwarezyklus

Dieser Abschnitt zeigt den Softwarezyklus von freier Software und welche Benutzer sich in den einzelnen Phasen an dem Projekt beteiligen bzw. wer die Software schließlich nutzt.

12.1 Entwickler

Man kann drei große Rollen in der freien Softwareprojekts unterscheiden. In der ersten Phase zieht es ausschließlich Entwickler an. Die Software ist noch nicht mehr als eine rohe Skizze, ein Entwurf dessen, was es verspricht, zu werden. Jeder kann sich die Software beschaffen, aber nur wer selbst am Quellcode mitarbeiten möchte, wird es tun.[25]

12.2 Power-User

In der zweiten Phase hat die Software ein Stabilität erlangt, die sie für *Power-User* (z. B. Systemadministratoren) interessant macht, die sie nicht primär weiterentwickeln, sondern einsetzen wollen. Installation, Integration und Wartung erfordern auch jetzt noch weiterreichende Programmierfähigkeiten, doch diese Nutzergruppe erhält dafür eine Funktionalität, Flexibilität und ein Maß an Kontrolle über die Software, die sie zu einer attraktiven Alternative zu proprietärer Software macht.

Hat die Software einen Grad an Stabilität erreicht, die sie für den praktischen Einsatz geeignet macht, gibt das Projekt ein offizielles Release heraus. Parallel zu dieser Produktionsversion wird an einer Entwicklerversion weitergearbeitet. Diese Zweiteilung erlaubt einerseits eine kontinuierliche Verbesserung und Erweiterung, an der sich alle Interessierten beteiligen können sowie andererseits das Einfrieren, Durchtesten und Stabilisieren von Momentaufnahmen aus diesem Prozess, die bis zum nächsten Release unverändert für diejenigen bleibt, die die Software in ihrer täglichen Arbeit verwenden wollen.[26]

12.3 Endnutzer

In der dritten Phase erhält die Software Merkmale wie Installationshilfen und Dokumentation, die sie schließlich auch für technisch weniger bedarfte Nutzer zugänglich macht. Dieser Schritt wird oft nicht mehr von den selbst motivierten Teilnehmern des freien Projekts durchgeführt (Coder schreiben keine Dokumentation), sondern von Dienstleistungsfirmen, die oft eigens für den Support

[25] Vgl. Grassmuck 2004, S. 254
[26] Vgl. Grassmuck 2004, S. 255

von freier Software gegründet wurden.

Auf dieser Stufe stellt sich auch die Frage einer Integration vom Betriebssystem über die grafische Benutzeroberfläche bis zu den Anwendungen. Zentralistischen Modellen von Unternehmen wie Apple oder Microsoft gelingt dies naturgemäß gut, ebenso einem enger gekoppelten Projektverbund wie GNU. In einer weniger eng gekoppelten Umgebung besteht dagegen ein Bedarf an Abstimmung, ohne dafür jedoch auf zentralisierte Gremien zurück fallen zu wollen.[27]

[27] Vgl. Grassmuck 2004, S. 255

13. Zusammenfassung

In der Arbeit wurde zuerst vorgestellt was ist freie Software und warum es sie gibt. Dann wurde in der Arbeit beschrieben was ist Quellcode und was der Objektcode ist. Danach ist gezeigt wie ein freies Software Projekt funktioniert und wer alles an diesem Projekt beteiligt ist. Zum Schluss wurde erklärt was ein Release ist mit welchen Werkzeugen gearbeitet wird und wie der Softwarezyklus aussieht. Durch die gute Qualität den großen Support der Community werden freie Software Projekte immer attraktiver auch für Firmen. Die schnelle Entwicklung garantiert eine schnelle Ausbesserung von Fehlern. Aber auch wie man an Wikipedia sieht ist eine Große Enzyklopädie durch die Community entstanden. Auch bei Software Projekten die durch Module aus der Community erweitert werden kann steigt die Funktionalität dieser Projekte. In der Zukunft wird sicher noch mehr freie Software zur Verfügung stehen das den Kommerziellen Produkten mehr und mehr Konkurrenz bietet.

14. Quellen

Internetverzeichnis:

(1)Bravehack, Jens Sieckmann,Url: http://www.bravehack.de/html/bravehack.html Zugriff 5 Aug 2009

(2) http://de.wikipedia.org/wiki/Freie_Software Zugriff 5 Aug 2009

(3)http://de.wikipedia.org/wiki/Wikipedia:Hauptseite Zugriff 5 Aug 2009

(4)Mr.Qoon;“Was ist Objekt-Code in einem Computer-Programm? “,Url: http://wasista.de/Internat/2009/0225/Was-ist-Objekt-Code-in-einem-Computer-Pr.html Zugriff 6 Aug 2009

(5)http://www2.dortmund.de/do4u_intern/artnet_archiv/ger/wor/grassm.html#N_19 Zugriff 6 Aug 2009

(6) http://de.wikipedia.org/wiki/Programmfehle Zugriff 6 Aug 2009

Literaturverzeichnis:

(1) Volker Grassmuck, Freie Software, 2. korrigierte Auflage, ISBN3-89331-569-1,2004